IBATIS - DYNAMIC SQL

Using dynamic queries is a very powerful feature of iBatis. Sometime you have changing WHERE clause criterion based on your parameter object's state. In such situation iBATIS provides a set of dynamic SQL tags that can be used within mapped statements to enhance the reusability and flexibility of the SQL.

All the logic is put in .XML file using some additional tags. Following is an example where SELECT statement would work in two ways:

- If you would pass an ID then it would return all the records corresponding to that ID,
- otherwise it would return all the records where employee ID is set to NULL.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sqlMap
PUBLIC "-//ibatis.apache.org//DTD SQL Map 2.0//EN"
"http://ibatis.apache.org/dtd/sql-map-2.dtd">

<sqlMap namespace="Employee">
<select >
   SELECT * FROM EMPLOYEE
   <dynamic prepend="WHERE ">
      <isNull property="id">
         id IS NULL
      </isNull>
      <isNotNull property="id">
         id = #id#
      </isNotNull>
   </dynamic>
</select>
</sqlMap>
```

You can check condition using <isNotEmpty> tag as follows. Here condition would be added only when passed property is not empty.

```
..................
<select >
   SELECT * FROM EMPLOYEE
   <dynamic prepend="WHERE ">
      <isNotEmpty property="id">
           id = #id#
      </isNotEmpty>
   </dynamic>
</select>
..................
```

If you wan a query where we can select on id and/or first name of an Employee. Here would be your SELECT statement:

```
..................
<select >
   SELECT * FROM EMPLOYEE
   <dynamic prepend="WHERE ">
      <isNotEmpty prepend="AND" property="id">
           id = #id#
      </isNotEmpty>
      <isNotEmpty prepend="OR" property="first_name">
           first_name = #first_name#
      </isNotEmpty>
   </dynamic>
</select>
..................
```

Example: Dynamic SQL

Following example would show how you can write SELECT statement with dynamic SQL. Consider, we have following EMPLOYEE table in MySQL:

```
CREATE TABLE EMPLOYEE (
   id INT NOT NULL auto_increment,
   first_name VARCHAR(20) default NULL,
   last_name  VARCHAR(20) default NULL,
   salary     INT  default NULL,
   PRIMARY KEY (id)
);
```

This table is having only one record as follows:

```
mysql> select * from EMPLOYEE;
+----+------------+-----------+--------+
| id | first_name | last_name | salary |
+----+------------+-----------+--------+
|  1 | Zara       | Ali       |   5000 |
|  2 | Roma       | Ali       |   3000 |
|  3 | Noha       | Ali       |   7000 |
+----+------------+-----------+--------+
3 row in set (0.00 sec)
```

Employee POJO Class:

To perform read operation, let us have Employee class in Employee.java file as follows:

```
public class Employee {
  private int id;
  private String first_name;
  private String last_name;
  private int salary;

  /* Define constructors for the Employee class. */
  public Employee() {}

  public Employee(String fname, String lname, int salary) {
    this.first_name = fname;
    this.last_name = lname;
    this.salary = salary;
  }

  /* Here are the method definitions */
  public int getId() {
    return id;
  }
  public String getFirstName() {
    return first_name;
  }
  public String getLastName() {
    return last_name;
  }
  public int getSalary() {
    return salary;
  }
} /* End of Employee */
```

Employee.xml File:

To define SQL mapping statement using iBATIS, we would add following modified <select> tag in Employee.xml file and inside this tag definition we would define an "id" which will be used in IbatisReadDy.java file for executing Dynamic SQL SELECT query on database.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sqlMap
PUBLIC "-//ibatis.apache.org//DTD SQL Map 2.0//EN"
"http://ibatis.apache.org/dtd/sql-map-2.dtd">

<sqlMap namespace="Employee">
<select >
   SELECT * FROM EMPLOYEE
   <dynamic prepend="WHERE ">
      <isNotNull property="id">
         id = #id#
      </isNotNull>
   </dynamic>
</select>
</sqlMap>
```

Above SELECT statement would work in two ways (i) If you would pass an ID then it would return records corresponding to that ID (ii) otherwise it would return all the records.

IbatisReadDy.java File:

This file would have application level logic to read conditional records from the Employee table:

```
import com.ibatis.common.resources.Resources;
import com.ibatis.sqlmap.client.SqlMapClient;
import com.ibatis.sqlmap.client.SqlMapClientBuilder;
import java.io.*;
import java.sql.SQLException;
import java.util.*;

public class IbatisReadDy{
  public static void main(String[] args)
   throws IOException,SQLException{
   Reader rd=Resources.getResourceAsReader("SqlMapConfig.xml");
   SqlMapClient smc=SqlMapClientBuilder.buildSqlMapClient(rd);

   /* This would read all records from the Employee table.*/
   System.out.println("Going to read records.....");
   Employee rec = new Employee();
   rec.setId(1);

   List <Employee> ems = (List<Employee>)
                    smc.queryForList("Employee.findByID", rec);
   Employee em = null;
   for (Employee e : ems) {
      System.out.print("  " + e.getId());
      System.out.print("  " + e.getFirstName());
      System.out.print("  " + e.getLastName());
      System.out.print("  " + e.getSalary());
      em = e;
      System.out.println("");
   }

   System.out.println("Records Read Successfully ");

  }
}
```

Compilation and Run:

Here are the steps to compile and run the above mentioned software. Make sure you have set PATH and CLASSPATH appropriately before proceeding for the compilation and execution.

- Create Employee.xml as shown above.
- Create Employee.java as shown above and compile it.

- Create IbatisReadDy.java as shown above and compile it.
- Execute IbatisReadDy binary to run the program.

You would get following result, and a record would be read from the EMPLOYEE table.

```
Going to read records.....
  1  Zara  Ali  5000
Record Reads Successfully
```

Try above example by passing **null** as *smc.queryForList("Employee.findByID", null)*.